图像复原基本算法

简介

对 lena 图像分别加入高斯噪声、椒盐噪声以及运动模糊，并分别利用均值去噪、中值去噪、逆滤波以及维纳滤波对图像进行复原，观察结果并使用 PSNR 和 SSIM 进行评价。

原理

(1) 均值去噪

设有一副$M \times M$的图像$f(x,y)$，若均值去噪后的图像为$g(x,y)$，则有：

$$g(x,y) = \frac{1}{N}\sum_{i,j \in s} f\ (i,j)$$

式中，$x,y = 0,1,\dots,M-1$，s为(x,y)领域内像素坐标的集合，N表示集合s内像素的总数

可见均值去噪就是将当前像素邻域内各像素的灰度平均值作为其输出值的去噪方法

(2) 中值去噪

设有一副$M \times M$的图像$f(x,y)$，若平滑图像为$g(x,y)$，则有：

$$g(x,y) = med\left[\sum_{i,j \in s} f\ (i,j)\right]$$

式中，$x,y = 0,1,\dots,M-1$，s为(x,y)领域内像素坐标的集合，med表示取集合s内像素的中位数

可见中值去噪就是将当前像素邻域内各像素的灰度中位数作为其输出值的去噪方法

(3) 逆滤波

根据线性位移不变系统图像的退化模型：

$$g(x,y) = f(x,y) * h(x,y) + n(x,y)$$

利用傅里叶变换从上式可得：

$$G(u,v) = F(u,v)H(u,v) + N(u,v)$$

则逆滤波复原的方式为：

$$F(u,v) = \frac{G(u,v)}{H(u,v)} - \frac{N(u,v)}{H(u,v)}$$

忽略噪声，上式可简化为：

$$F(u,v) = \frac{G(u,v)}{H(u,v)}$$

(4) 维纳滤波

维纳滤波复原的表达式为：

$$F\hat{}(u,v) = [\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_{nn}(u,v)/S_{ff}(u,v)}]G(u,v)$$

其中，$H(u,v)$为退化函数，$S_{nn}(u,v) = |N(u,v)|^2$为噪声的功率谱，$S_{ff}(u,v) = |F(u,v)|^2$为为退化图像的功率谱。

MATLAB 代码

```
clc;clear;
image = rescale(double(imread('lena512.bmp')));
[m, n] = size(image);
figure(1);imshow(image, []);title('原图像')



%%
%添加高斯噪声并复原
imageNoise1 = imnoise(image, 'gaussian');

%M*N 邻域平滑
M1 = 3;
N1 = 3;

%MATLAB 自带函数实现均值去噪
imageDenoise1 = imfilter(imageNoise1, fspecial('average', [M1, N1]));

%%代码实现均值去噪
% imageDenoise1 = image;
% for i = (M1 + 1) / 2:m - (M1 - 1)
%     for j = (N1 + 1) / 2:n - (N1 - 1)
%         imageDenoise1(i, j) = mean(imageNoise1(i - (M1 - 1) / 2:i + (M1 - 1) / 2,...
%             j - (N1 - 1) / 2:j + (N1 - 1) / 2), 'all');
%         imageDenoise1(i, j) = mean(imageNoise1(i - (M1 - 1) / 2:i + (M1 - 1) / 2,...
%             j - (N1 - 1) / 2:j + (N1 - 1) / 2), 'all');
%     end
% end

figure(2);
subplot(1, 2, 1);imshow(imageNoise1, []); title('加高斯噪声');
subplot(1, 2, 2);imshow(imageDenoise1, []); title([num2str(M1), '*', num2str(N1), '邻域平滑']);

disp(newline);
```

```
disp(['加高斯噪声后', num2str(M1), '*', num2str(N1), '邻域平滑']);
disp(['PSNR:', num2str(imPSNR(image, imageDenoise1))]);
disp(['SSIM:', num2str(imSSIM(image, imageDenoise1))]);



%%
%添加椒盐噪声并复原
imageNoise2 = imnoise(image, 'salt & pepper');

%M*N 窗口中值滤波
M2 = 3;
N2 = 3;

%MATLAB 自带函数实现中值去噪
imageDenoise2 = medfilt2(imageNoise2, [M2, N2]);

%%代码实现中值去噪
% imageDenoise2 = image;
% for i = (M2 + 1) / 2:m - (M2 - 1)
%         for j = (N2 + 1) / 2:n - (N2 - 1)
%             imageDenoise2(i, j) = median(imageNoise2(i - (M2 - 1) / 2:i + (M2 - 1) / 2,...
%                 j - (N2 - 1) / 2:j + (N2 - 1) / 2), 'all');
%             imageDenoise2(i, j) = median(imageNoise2(i - (M2 - 1) / 2:i + (M2 - 1) / 2,...
%                 j - (N2 - 1) / 2:j + (N2 - 1) / 2), 'all');
%         end
% end

figure(3);
subplot(1, 2, 1);imshow(imageNoise2, []); title('加椒盐噪声');
subplot(1, 2, 2);imshow(imageDenoise2, []); title([num2str(M2), '*', num2str(N2), '窗口中值去
噪']);

disp(newline);
disp(['加椒盐噪声后', num2str(M2), '*', num2str(N2), '窗口中值去噪']);
disp(['PSNR:', num2str(imPSNR(image, imageDenoise2))]);
disp(['SSIM:', num2str(imSSIM(image, imageDenoise2))]);



%%
%添加运动模糊并复原
blur = fspecial('motion', 25, 11);
imageBlur = imfilter(image, blur, 'circular');
```

```
%逆滤波
blurFft1 = fft2(blur, m, n);
imageBlurFft1 = fft2(imageBlur);
imageDeblurFft1 = imageBlurFft1 ./ blurFft1;
imageDeblur1 = ifft2(imageDeblurFft1);
figure(4);
subplot(1, 2, 1);imshow(imageBlur, []);title('加运动模糊');
subplot(1, 2, 2);imshow(imageDeblur1, []);title('直接逆滤波');

disp(newline);
disp('加运动模糊后直接逆滤波');
disp(['PSNR:', num2str(imPSNR(image, imageDeblur1))]);
disp(['SSIM:', num2str(imSSIM(image, imageDeblur1))]);


%添加噪声后逆滤波
imageBlurNoise = imnoise(imageBlur, 'gaussian');
blurFft2 = fft2(blur, m, n);
imageBlurFft2 = fft2(imageBlurNoise);
imageDeblurFft2 = imageBlurFft2 ./ blurFft2;
imageDeblur2 = ifft2(imageDeblurFft2);
figure(5);
subplot(1, 2, 1);imshow(imageBlurNoise, []);title('加运动模糊及高斯噪声');
subplot(1, 2, 2);imshow(imageDeblur2, []);title('直接逆滤波');

disp(newline);
disp('加运动模糊及高斯噪声后直接逆滤波');
disp(['PSNR:', num2str(imPSNR(image, imageDeblur2))]);
disp(['SSIM:', num2str(imSSIM(image, imageDeblur2))]);


%维纳滤波
imageDeblur3 = deconvwnr(imageBlur, blur, 0);
figure(6);
subplot(1, 2, 1);imshow(imageBlur, []);title('加运动模糊');
subplot(1, 2, 2);imshow(imageDeblur3, []);title('维纳滤波');

disp(newline);
disp('加运动模糊后维纳滤波（不带参数）');
disp(['PSNR:', num2str(imPSNR(image, imageDeblur3))]);
disp(['SSIM:', num2str(imSSIM(image, imageDeblur3))]);
```

```
%添加噪声后维纳滤波（噪信比）
nsr = sum((image - imageBlurNoise) .^ 2, 'all') ./ sum(image .^ 2, 'all');
imageDeblur4 = deconvwnr(imageBlurNoise, blur, nsr);
figure(7);
subplot(1, 2, 1);imshow(imageBlurNoise, []);title('加运动模糊及高斯噪声');
subplot(1, 2, 2);imshow(imageDeblur4, []);title('加入噪信比维纳滤波');

disp(newline);
disp('加运动模糊及高斯噪声后维纳滤波（噪信比）');
disp(['PSNR:', num2str(imPSNR(image, imageDeblur4))]);
disp(['SSIM:', num2str(imSSIM(image, imageDeblur4))]);


%添加噪声后维纳滤波（自相关函数）
ncorr = abs(ifft2(abs(fft2((imageBlurNoise - image))) .^ 2));
icorr = abs(ifft2(abs(fft2((image))) .^ 2));
imageDeblur5 = deconvwnr(imageBlurNoise, blur, ncorr, icorr);
figure(8);
subplot(1, 2, 1);imshow(imageBlurNoise, []);title('加运动模糊及高斯噪声');
subplot(1, 2, 2);imshow(imageDeblur5, []);title('加入自相关函数维纳滤波');

disp(newline);
disp('加运动模糊及高斯噪声后维纳滤波（自相关函数）');
disp(['PSNR:', num2str(imPSNR(image, imageDeblur5))]);
disp(['SSIM:', num2str(imSSIM(image, imageDeblur5))]);
```

结果分析

首先，原图使用 lena 图像

原图像

加入高斯噪声并使用 3*3 邻域平滑均值去噪

加高斯噪声

3*3邻域平滑

加高斯噪声后3*3邻域平滑
PSNR: 75.4644
SSIM: 0.99997

加入椒盐噪声并使用 3*3 窗口中值去噪

加椒盐噪声

3*3窗口中值去噪

加椒盐噪声后3*3窗口中值去噪
PSNR: 81.1168
SSIM: 0.99999

加入运动模糊后直接逆滤波

加运动模糊

直接逆滤波

加运动模糊后直接逆滤波

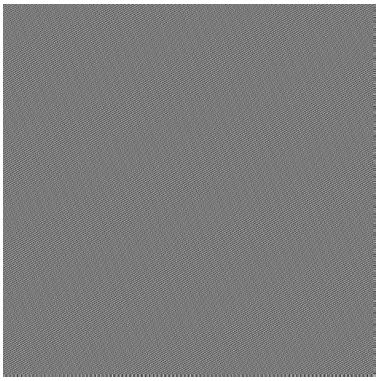
PSNR: 61.7458

SSIM: 0.99926

加入运动模糊及高斯噪声后直接逆滤波

加运动模糊及高斯噪声

直接逆滤波



加运动模糊及高斯噪声后直接逆滤波
PSNR:-7.6649
SSIM:0.00015425
可见逆滤波对噪声非常敏感

加入运动模糊后维纳滤波（不带参数）

加运动模糊

维纳滤波

加运动模糊后维纳滤波（不带参数）
PSNR:104.8433
SSIM:1

加入运动模糊及高斯噪声后维纳滤波（带噪信比参数）

加运动模糊及高斯噪声

加入噪信比维纳滤波

加运动模糊及高斯噪声后维纳滤波（噪信比）

PSNR: 67.4198

SSIM: 0.99968

加入运动模糊及高斯噪声后维纳滤波（带自相关函数参数）

加运动模糊及高斯噪声

加入自相关函数维纳滤波

加运动模糊及高斯噪声后维纳滤波（自相关函数）

PSNR:69.3723

SSIM:0.99987